❑ Write a XML code to store following information about Book

- Library has multiple books with single book
- Single book have ISBN attribute
- Book also have name,author, page and price element
- Books can be written by multiple author with author_name, email
- Price have currency attribute with NPR and usd choice with default NPR value

Xml file:

```
<?xml version="1.0"?>
<!DOCTYPE books SYSTEM "book1.dtd">
<books>
        <book isbn="1234">
            <name>OS</name>
            <author>
            <author_name>Peter</author_name>
            <email>peter@gmail.com</email>
            </author>
            <page>150</page>
            <price currency="NPR">500</price>
            </book>
    <book isbn="2345">
            <name>Web Tech.</name>
            <author>
            <author_name>James</author_name>
            <email>jam@gmail.com</email>
            </author>
<author>
<author_name>Pop</author_name>
<email>pp@gmail.com</email>
</author>
<page>300</page>
<price currency="NPR">1200</price>
</book>
```

```
<book isbn="1111">
        <name>HTML</name>
        <author>
        <author_name>Bob</author_name>
        <email>bob@gmail.com</email>
        </author>
        <page>250</page>
        <price currency="usd">50</price>
        </book>
</books>
```

DTD file:

```
<!ELEMENT books(book+)>

<!ELEMENT book(name,author+,page,price)>

<!ELEMENT name(#PCDATA)>

<!ELEMENT author(author_name,email)>

<!ELEMENT author_name(#PCDATA)>

<!ELEMENT email(#PCDATA)>

<!ELEMENT page(#PCDATA)>

<!ELEMENT price(#PCDATA)>

<!ATTLIST book isbn CDATA #REQUIRED>

<!ATTLIST price currency(NPR|usd) “NPR”>
```

2B.xml

C: > xampp > htdocs > bca > 2B.xml

```xml
<?xml version="1.0"?>
<!DOCTYPE books SYSTEM "book1.dtd">
<books>
        <book isbn="1234">
    <name>OS</name>
    <author>
    <author_name>Peter</author_name>
    <email>peter@gmail.com</email>
    </author>
    <page>150</page>
    <price currency="NPR">500</price>
    </book>
     <book isbn="2345">
    <name>Web Tech.</name>
    <author>
    <author_name>James</author_name>
    <email>jam@gmail.com</email>
    </author>
<author>
<author_name>Pop</author_name>
<email>pp@gmail.com</email>
</author>
<page>300</page>
<price currency="NPR">1200</price>
```

xml file:

2B.xml

C: > xampp > htdocs > bca > 2B.xml

```xml
<?xml version="1.0"?>
<!DOCTYPE books SYSTEM "book1.dtd">
<books>
        <book isbn="1234">
    <name>OS</name>
    <author>
    <author_name>Peter</author_name>
    <email>peter@gmail.com</email>
    </author>
    <page>150</page>
    <price currency="NPR">500</price>
    </book>
     <book isbn="2345">
    <name>Web Tech.</name>
    <author>
    <author_name>James</author_name>
    <email>jam@gmail.com</email>
    </author>
<author>
<author_name>Pop</author_name>
<email>pp@gmail.com</email>
</author>
<page>300</page>
<price currency="NPR">1200</price>
</book>
<book isbn="1111">
    <name>HTML</name>
    <author>
    <author_name>Bob</author_name>
    <email>bob@gmail.com</email>
    </author>
    <page>250</page>
    <price currency="usd">50</price>
    </book>
</books>

```

dtd file:

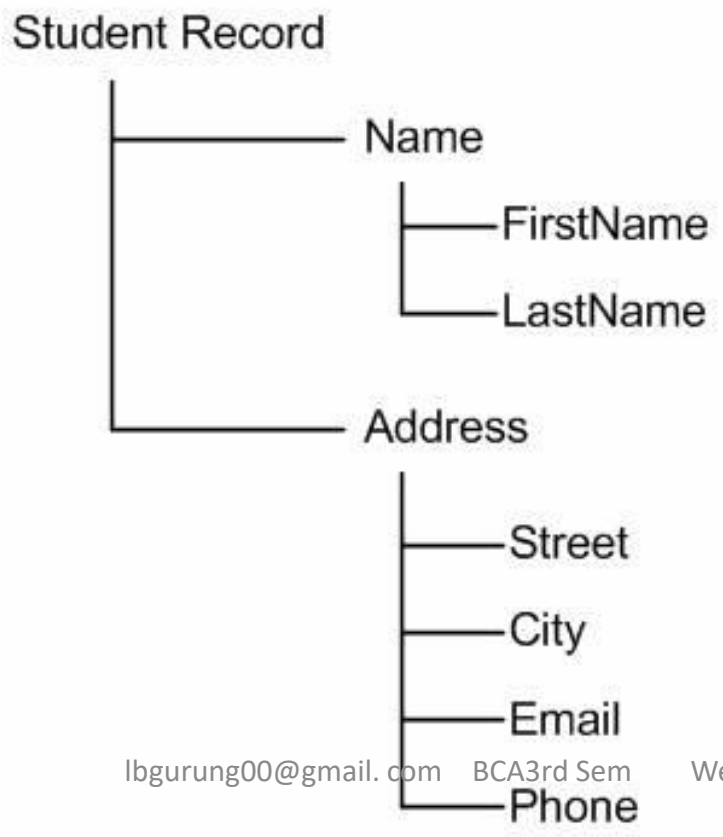
C: > xampp > htdocs > bca > book1.dtd

```
1    <!ELEMENT books(book+)>
2    <!ELEMENT book(name,author+,page,price)>
3    <!ELEMENT name(#PCDATA)>
4    <!ELEMENT author(author_name,email)>
5    <!ELEMENT author_name(#PCDATA)>
6    <!ELEMENT email(#PCDATA)>
7    <!ELEMENT page(#PCDATA)>
8    <!ELEMENT price(#PCDATA)>
9    <!ATTLIST book isbn CDATA #REQUIRED>
10   <!ATTLIST price currency(NPR|usd) "NPR">
11
```

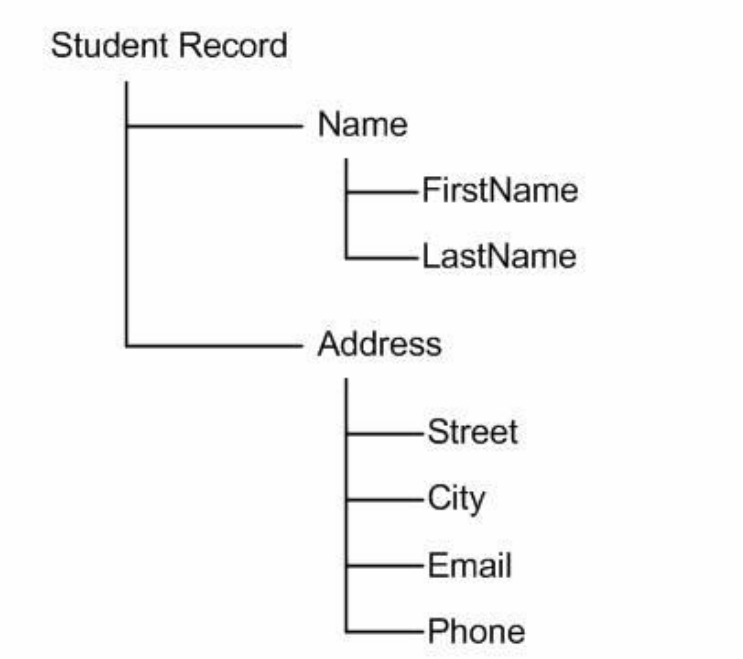
Use XML to create a student record database for the student information management.

- Each student record includes student's name and address.
- The name has two parts: First name and last name.
- The address has four parts: Street, City, Email and Phone Number

The structure of the student record can be presented as:



```
<?xml version = "1.0"?>
<!--student1.xml-->
<students>
   <student>
    <name>
        <firstname> Peter </firstname>
        <lastname> Baral </lastname>
</name>
<address>
       <street> 101 South Street</street>
       <city> Brt </city>
        <email> pb@gmail.com </email>
         <phone> 253456 </phone>
</address>
</student>
<student>
     <name>
        <firstname> Tom </firstname>
         <lastname> Shrestha </lastname>
    </name>
<address>
     <street> 202 College Road </street>
      <city> Brt </city>
     <email> tom@gmail.com</email>
       <phone> 2213456 </phone>
</address>
</student>
</students>
```



DTD file

```
<?xml version = "1.0"?>
<!--students.dtd-a document type definition for the students.xml-->
<!ELEMENT students (student+)>
<!ELEMENT student (name,address)>
<!ELEMENT name (firstname,lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT address (street,city,email,phone)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT Phone(#PCDATA)>
```

Note: To use the DTD file, we must add this code into the XML file.

```
<!DOCTYPE students SYSTEM "students.dtd">
```

Note: To use the DTD file, we must add this code into the XML file.

<!DOCTYPE students SYSTEM "students.dtd">

```
<?xml version = "1.0"?>
<!DOCTYPE students SYSTEM "students.dtd">
   <student>
    <name>
         <firstname> Peter </firstname>
         <lastname> Baral </lastname>
</name>
<address>
       <street> 101 South Street</street>
       <city> Brt </city>
        <email> pb@gmail.com </email>
         <phone> 253456 </phone>
</address>
</student>
<student>
      <name>
         <firstname> Tom </firstname>
          <lastname> Shrestha </lastname>
    </name>
<address>
      <street> 202 College Road </street>
        <city> Brt </city>
       <email> tom@gmail.com</email>
         <phone> 2213456 </phone>
</address>
</student>
</students>
```

Display the XML file in HTML by CSS and XSLT

```
<!--students.css- a style sheet for the students.xml document-->
student{display: block; margin-top: 15px; color: blue;}
name{display: block; margin-left:40px;margin-top: 30pt;color:red}
firstname{font-size: 28pt;}
lastname{font-size: 28pt;}
address{display: block; margin-left:40px;color:green}
street{display: block;font-size: 18pt;}
city{display: block;font-size: 18pt;}
email{display: block;font-size: 18pt;color:blue}
phone{display: block;font-size: 18pt;}
```

To use the CSS file, the XML must add the following sentence:

```
<?xml-stylesheet type="text/css" href= "students.css"?>
```

XSD Indicators

- The indicators used to control the elements presentation in the documents.
- We can control how elements are to be used in documents with indicators.
- There are seven different indicators classified into 3 types:

There are seven indicators:

- Order indicators:
  1. All
  2. Choice
  3. Sequence
- Occurrence indicators:
  4. maxOccurs
  5. minOccurs
- Group indicators:
  6 Group name
  7. attributeGroup name

Types of XSD Indicators

1. Order Indicators

It defines the order of the element

- ✓ **All** – Child elements can occur in any order.
- ✓ **Choice** – Only one of the child element can occur.
- ✓ **Sequence** – Child element can occur only in specified order.

Types of XSD Indicators

1. Order Indicators

Example:

```
<xs:element name="person">
  <xs:complexType>
    <xs:all>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Types of XSD

Indicators

1. Order Indicators

Example:

```
<xs:element name = "book" >
<xs:complexType>
<xs:choice>
 <xs:element name = "company" type = "xs:string" />
<xs:element name = "employe" type = "xs:integer" />
</xs:choice>
</xs:complexType>
</xs:element>
```

Types of XSD

Indicators

1. Order Indicators

Example:

```
<xs:element name = "book" >
<xs:complexType>
<xs:sequence>
 <xs:element name = "company" type = "xs:string" />
<xs:element name = "type" type = "xs:string" />
</xs:sequence>
</xs:complexType>
</xs:element>
```

Types of XSD Indicators

2. Occurrence Indicators:

It defines the number of times an element can occur.

- ✓ **maxOccurs** – Child element can occur only maxOccurs number of times.
- ✓ **minOccurs** – Child element must occur minOccurs number of times.

Types of XSD Indicators

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string" maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The example above indicates that the "child_name" element can occur a minimum of one time (the default value for minOccurs is 1) and a maximum of ten times in the "person" element.

Types of XSD Indicators

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
      maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The example above indicates that the "child_name" element can occur a minimum of zero times and a maximum of ten times in the "person" element.

Types of XSD Indicators

3. Group Indicators:

Group indicators define related sets of elements.

- ✓ **Group** – Defines related set of elements.
- ✓ **attributeGroup** – Defines related set of attributes.

Types of XSD Indicators

3. Group Indicators:

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>
```

lbgurung00@gmail.com

Types of XSD Indicators

3. Group Indicators:

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>
```

❑ Difference between DTD and XSD

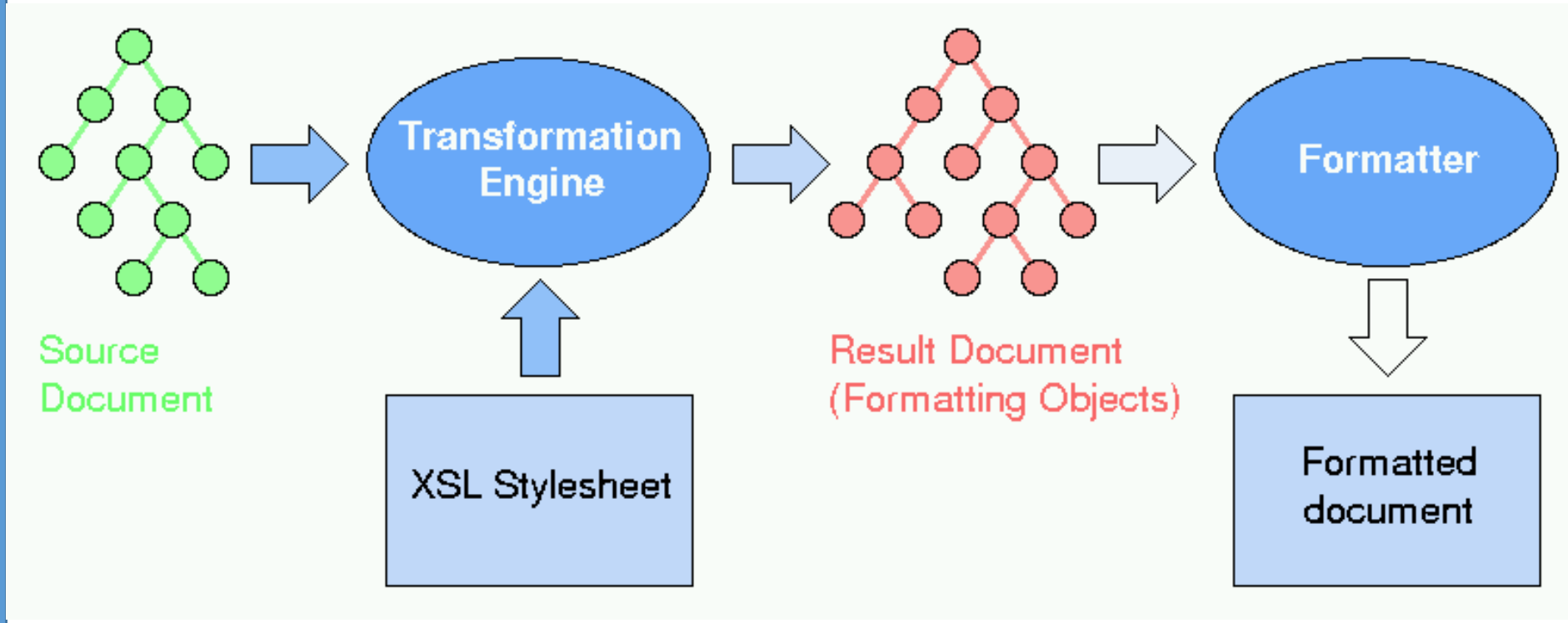
DTD	XSD
1. It stands for Document Type Definition.	1. It stands for XML Schema Definition.
2. DTD does not support data type	2. XSD supports data type
3. DTD does not support namespace	3. XSD supports namespace
4. DTD does not define order for child elements.	4. XSD defines order for child elements
5. DTD is not extensible.	5. XSD is extensible.
6. DTDs are derived from SGML syntax	6. XSDs are written in XML
7. In DTD we can't specify restriction on data	7. In XSD, we can specify restriction on data
8. Example: <!DOCTYPE Address [<!ELEMENT Address(name)> <!ELEMENT name(#PCDATA)>]>	8. Example: <xs:element name ="Address"> <xs:complextype> <xs:sequence> <xs:element name ="name" type="xs:string" /> </xs:sequence> </xs:complextype> </xs:element>

❑ XSL/XSLT

- XSL(E**X**tensible **S**tylesheet **L**anguage)
  - ✓ It is a style sheet language for XML document
  - ✓ XSL describes how the XML document should be displayed

lbgurung00@gmail.com

- XSL Architecture



- XSL Components

XSL is constituted of three main components:

- ✓ XSLT: a language for transforming XML document
- ✓ XPath: A language for navigating in XML documents
- ✓ XSL-FO: a language for formatting XML document

lbgurung00@gmail.com

- XSLT(XSL Transformation)

- ✓ XSLT transforms an XML document into another XML document, or another type of document that is recognized by a browser like HTML or XHTML.
- ✓ XSLT uses a Xpath to navigate in XML documents
- ✓ With XSLT you can add/remove elements and attriute to or from the output side.
- ✓ With XSLT you can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display etc.
- ✓ XSLT is a part of XSL(eXtensible stylesheet language) and is written within XSL document whose file extension is .xsl

Example:

In this example, creating the XML file that contains the information about five students and displaying the XML file using XSLT.

XML file:

Creating bca.xml as:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="mmamc.xsl"?>
 <student>
  <s>
   <name> Khem  </name>
   <branch> CSE</branch>
   <age>18</age>
   <city> Brt </city>
  </s>
  <s>
   <name> Anish  </name>
   <branch> CSE</branch>
   <age> 20</age>
   <city> KTM </city>
  </s>
  <s>
   <name> Simran </name>
   <branch> CSE</branch>
   <age> 23</age>
   <city> Pokhara</city>
  </s>
  <s>
   <name> Basant </name>
   <branch> CSE</branch>
   <age> 17</age>
   <city> Dharan</city>
  </s>
  <s>
   <name> Mahesh </name>
   <branch> IT</branch>
   <age> 25</age>
   <city> Ithari</city>
  </s>
 </student>
```

In this example, bca.xml is created and linking it with mmamc.xsl which contains the corresponding XSL style sheet rules.

XSLT Code:

Creating mmamc.xsl as:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
  <h1 align="center">Students' Basic Details</h1>
   <table border="3" align="center" >
   <tr>
    <th>Name</th>
    <th>Branch</th>
    <th>Age</th>
    <th>City</th>
   </tr>
    <xsl:for-each select="student/s">
   <tr>
    <td><xsl:value-of select="name"/></td>
    <td><xsl:value-of select="branch"/></td>
    <td><xsl:value-of select="age"/></td>
    <td><xsl:value-of select="city"/></td>
   </tr>
    </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Output :

Students' Basic Details

Name	Branch	Age	City
Khem Baral	CSE	18	Brt
Anish	CSE	20	KTM
Simran	CSE	23	Pokhara
Basant	CSE	17	Dharan
Mahesh	IT	25	Ithari

Example:
Save this code as store.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="example.xsl"?>
<bookstore>
<book>
<title>Let us C</title>
<Author>Yashwant</Author>
</book>
<book>
<title>Operating system</title>
<Author>Peter</Author>
</book>
</bookstore>
```

Save this code as example.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0">
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h1>Book Collection</h1>
<table border="1" cellpadding="10" cellspacing="0">
 <tr>
   <th>Title</th>
            <th>Author</th>
            </tr>
            <xsl:for-each select="bookstore/book">
            <tr>
            <td><xsl:value-of select="title"/></td>
            <td><xsl:value-of select="Author"/></td>
            </tr>
            </xsl:for-each>
            </table>
            </body>
            </html>
            </xsl:template>
            </xsl:stylesheet>
```

Output :

Book Collection

Title	Author
Let us C	Yashwant
Operating system	Peter

Display the XML file in HTML by CSS and XSLT

```
<?xml version = "1.0"?>
<?xml-stylesheet type="text/css" href= "students.css"?>
   <student>
     <name>
          <firstname> Peter </firstname>
          <lastname> Baral </lastname>
</name>
<address>
        <street> 101 South Street</street>
        <city> Brt </city>
         <email> pb@gmail.com </email>
          <phone> 253456 </phone>
</address>
</student>
<student>
      <name>
          <firstname> Tom </firstname>
           <lastname> Shrestha </lastname>
    </name>
<address>
       <street> 202 College Road </street>
         <city> Brt </city>
       <email> tom@gmail.com</email>
          <phone> 2213456 </phone>
</address>
</student>
</students>
```

XSLT

To use the XSLT file, we must add the following sentence into XML file.

<?xml-stylesheet type = "text/xsl" href = "students.xsl"?>

```
<?xml version ="1.0"?>
<!--students.xsl-->
<xsl:stylesheet version = "1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/TR/xhtml1/strict">
<xsl:template match = "/">
<h2> Student Record </h2>
<xsl:for-each select="students/student"> <br/>
<span style="font-weight:bold;color:red"> FirstName: </span>
<xsl:value-of select="name/firstname" />
<span style="font-weight:bold;color:red"> LastName: </span>
<xsl:value-of select="name/lastname" /> <br/>
<span style="font-weight:bold;color:green"> Street: </span>
<xsl:value-of select="address/street" /> <br/>
<span style="font-weight:bold;color:green"> City: </span>
<xsl:value-of select="address/city" /> <br/>
<span style="font-weight:bold;color:blue"> Email: </span>
<xsl:value-of select="address/email" /> <br/>
<span style="font-weight:bold;color:green"> Phone: </span>
<xsl:value-of select="address/phone" /> <br/>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

XSLT

To use the XSLT file, we must add the following sentence into XML file. <?xml-stylesheet type = "text/xsl" href = "students.xsl"?>

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "students.xsl"?>
   <student>
     <name>
          <firstname> Peter </firstname>
          <lastname> Baral </lastname>
</name>
<address>
        <street> 101 South Street</street>
        <city> Brt </city>
         <email> pb@gmail.com </email>
          <phone> 253456 </phone>
</address>
</student>
<student>
      <name>
          <firstname> Tom </firstname>
           <lastname> Shrestha </lastname>
    </name>
<address>
      <street> 202 College Road </street>
        <city> Brt </city>
       <email> tom@gmail.com</email>
          <phone> 2213456 </phone>
</address>
</student>
</students>
```

Xquery

Introduction:

- XQuery is ***an expression*** language for querying XML data
- It was designed to query XML data.
- XQuery for XML is like SQL for databases
- XQuery is built on XPath expressions
- XQuery is supported by all major databases
- XQuery is a W3C Recommendation

Use of Xquery

XQuery can be used to

- Extract information to use in a web service
- Generate summary reports
- transform XML data to XHTML
- Search web documents to relevant information

Advantages of Xquery

Following are the advantages of Xquery:

- ✓ It retrieves both hierarchal and tabular data
- ✓ It can be used to build webpages
- ✓ XQuery can be used to query web pages.
- ✓ XQuery is best for XML-based databases and object-based databases. Object databases are much more flexible and powerful than purely tabular databases.
- ✓ XQuery can be used to transform XML documents into XHTML documents.

Syntax Rules For Xquery

- ✓ XQuery is case sensitive.
- ✓ Attributes and variables that we used should have valid XML names in XQuery elements.
- ✓ XQuery should be written within single and double quotes.
- ✓ XQuery variables should be defined as: $(variable name). For example: $book.
- ✓ XQuery comment can be written between colon (:). like :(: Comment in XQuery:)

What Do You Mean By Xquery Flwor?

FLWOR is an acronym which stands for "For, Let, Where, Order by, Return".

For - It is used to select a sequence of nodes.

Let - It is used to bind a sequence to a variable.

Where - It is used to filter the nodes.

Order by - It is used to sort the nodes.

Return - It is used to specify what to return (gets evaluated once for every node).

Xpath:

- ✓ XPath (the XML Path language) is a language for finding information in an XML document.
- ✓ XPath is a syntax for defining parts of an XML document
- ✓ XPath uses path expressions to navigate in XML documents
- ✓ XPath contains a library of standard functions
- ✓ XPath is a major element in XSLT
- ✓ XPath is also used in XQuery, XPointer and Xlink
- ✓ XPath is a W3C recommendation
- ✓ XPath uses a path expression to select node or a list of nodes from an XML document.

Xpath:

Note:

Today XPath expressions can also be used in JavaScript, Java, XML Schema, PHP, Python, C and C++, and lots of other languages.

For example:

Xpath expression:

/bookstore/book[1] → Selects the first book element that is the child of the bookstore element

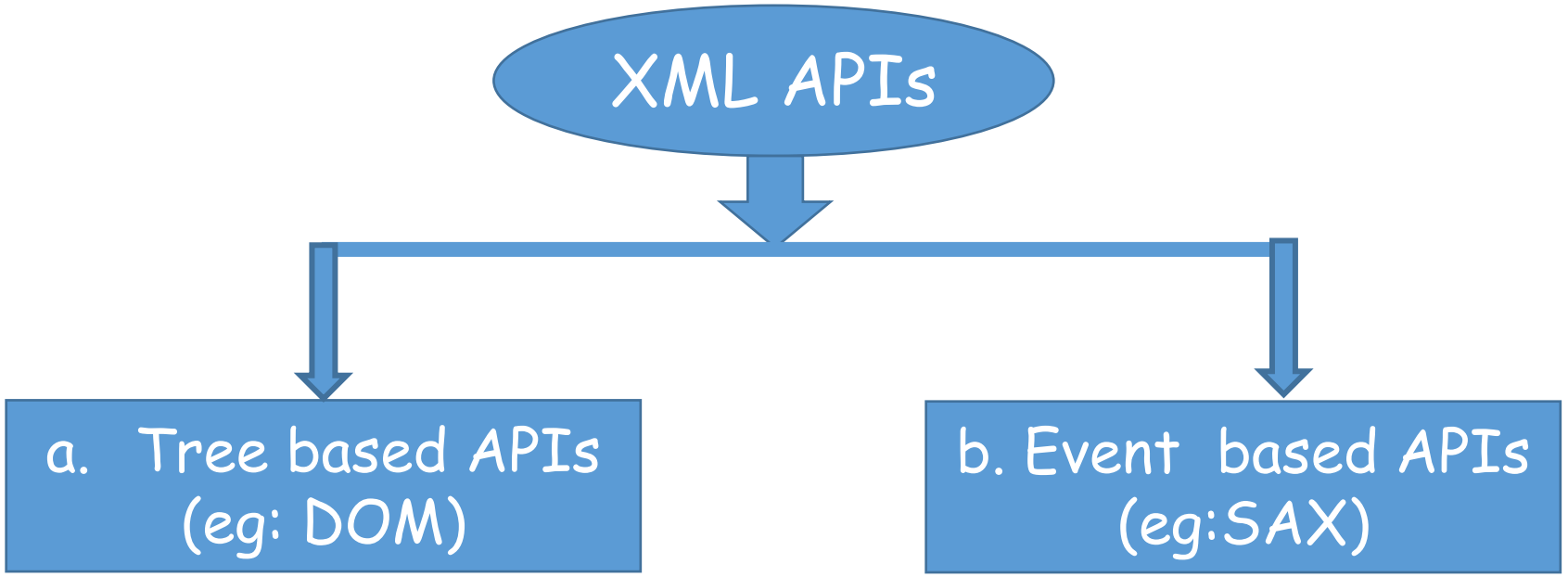
/bookstore/book[price>35.00] → Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00

- **Xquery vs Xpath**

Xquery	Xpath
1. XQuery is a functional programming and query language that is used to query a group of XML data.	1. XPath is a xml path language that is used to select nodes from an xml document using queries.
2. XQuery is used to extract and manipulate data from either xml documents or relational databases and ms office documents that support an xml data source	2. XPath is used to compute values like strings, numbers and boolean types from another xml documents.
3. xquery supports xpath and extended relational models.	3. xpath is still a component of query language.
4. xquery language helps to create syntax for new xml documents.	4. xpath was created to define a common syntax and behavior model for xpointer and xslt.

XML APIs

Two major types of XML APIs



- Compiles an XML document into an internal tree strucutre , then allows an application to navigate that tree.

Example: DOM

- Reports parsing events directly to the application through callbacks
- Usually does not build an internal tree

Example : SAX

SAX(Simple API for XML)

- ✓ It is an event – driven parser for XML documents, with an API interface.
- ✓ SAX provides a mechanism for reading data from an XML document.
- ✓ Originally designed as Java API but now supported by C++,Python , Perl etc.

SAX Features

- ✓ **Event driven:** user provider various event handlers
- ✓ **Fast and lightweight**: Document does not have to be entirely in memory
- ✓ **Sequential read access only**: Does not support modification of document

DOM(Document Object Model)

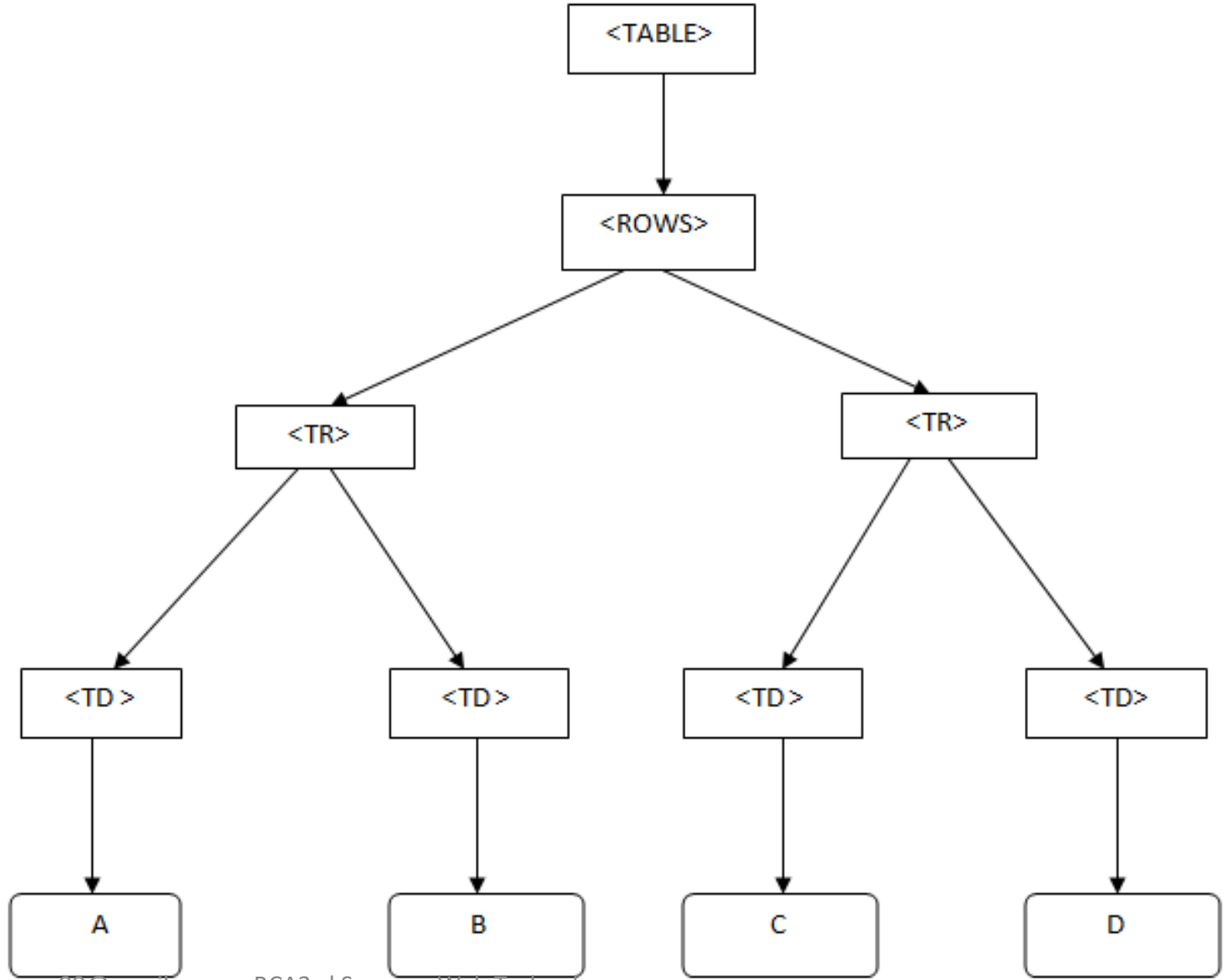
- ✓ DOM is an acronym stands for Document Object Model.
- ✓ It defines a standard way to access and manipulate documents.
- ✓ The Document Object Model (DOM) is a programming API for HTML and XML documents.
- ✓ It defines the logical structure of documents and the way a document is accessed and manipulated
- ✓ XML DOM defines a standard way to access and manipulate XML documents.
- ✓ The XML DOM makes a tree-structure view for an XML document.

DOM(Document Object Model)

- For example, consider this table,

```
<TABLE>
<ROWS>
<TR>
<TD>A</TD>
<TD>B</TD>
</TR>
<TR>
<TD>C</TD>
<TD>D</TD>
</TR>
</ROWS>
</TABLE>
```

The Document Object Model represents this table like this:



❑SAX vs DOM

SAX	DOM
1. It stands for Simple API for XML	1. It stands for Document Object Model
2. SAX is event driven based	2. DOM is tree model
3. Parses node by node	3. Stores the entire XML document into memory before processing
4. We cant insert or delete a node	4. We can insert or delete nodes
5. doesn't preserve comments	5. Preserve comments
6. Doesn't store the XML in memory	6. Occupies more memory
7. Top to bottom traversing	7. Traverse in any direction.
8. runs a little faster than DOM	8. Runs a bit slow than SAX

❑ Advantages of DOM

- ✓ XML DOM is language and platform independent
- ✓ XML DOM structure is modifiable, and values can be added, changed and removed
- ✓ XML DOM structure is traversable, and it can be randomly accessed by traversing the tree.
- ✓ Allows developer to navigate around the hierarchy

Disadvantages of DOM

- ✓ Consumes more memory(structure is very large)
- ✓ Programs written once remains in memory all the time
- ✓ Runs slower as compared to SAX(use of large memory)

❑ What are static and dynamic Web pages?

Web pages can be either static or dynamic. "Static" means unchanged or constant, while "dynamic" means changing or lively. Therefore, static Web pages contain the same prebuilt content each time the page is loaded, while the content of dynamic Web pages can be generated on-the-fly.

Standard HTML pages are static Web pages. They contain HTML code, which defines the structure and content of the Web page. Each time an HTML page is loaded, it looks the same. The only way the content of an HTML page will change is if the Web developer updates and publishes the file.

Other types of Web pages, such as PHP, ASP, and JSP pages are dynamic Web pages. These pages contain "server-side" code, which allows the server to generate unique content each time the page is loaded. For example, the server may display the current time and date on the Web page. It may also output a unique response based on a Web form the user filled out. Many dynamic pages use server-side code to access database information, which enables the page's content to be generated from information stored in the database. Websites that generate Web pages from database information are often called database-driven websites.

User can often tell if a page is static or dynamic simply by looking at the page's file extension in the URL, located in the address field of the Web browser. If it is ".htm" or ".html," the page is probably static. If the extension is ".php," ".asp," or ".jsp," the page is most likely dynamic.

Cookies and Session

lbgurung00@gmail.com

What is Cookie?

- Cookies are text files stored on the client computer that contains user information.
- A cookie is often used to identify a user.
- Cookies may contain: lbgurung00@gmail.com

  - ✓ Username and Password
  - ✓ Links that you've clicked on
  - ✓ Your cart information for buying online
  - ✓ Visitor tracking information
  - ✓ Your computers general location in the world
  - ✓ Video's you've watched

Uses of Cookies

Most popular uses of cookies

- To store username/password information so user doesn't have to log in every time ("remember me").
- To simply remember the user's name.
- To keep track of a users progress during a specified process.
- To remember a users theme.

lbgurung00@gmail.com

Types of Cookies

There are two types of cookie:

a. Non-persistent cookie
b. Persistent cookie

Non-persistent cookie:

It is valid for single session only. It is removed each time when user closes the bowser.

Persistent cookie:

It is valid for multiple session. It is not removed each time when user closes browser. It is removed only if user logout or signout

Advantages of Cookies

- Simplest technique of maintaining the state
- Cookies are maintained at client side
- The cookies make browsing the Internet faster & easier
- The cookies are stored on your hard drive in the text file under cookie.txt , You can view or edit & delete them .

lbgurung00@gmail.com

Difference between cookies and Sessions

Cookies	Sessions
1. Cookies are client-side text files that contains user information.	1. Sessions are Server-side files that contains user information.
2. Cookies are stored in the user's browser.	2. Sessions are not stored in user's browser.
3. A cookie can keep information in the user's browser until deleted	3. Session is that when you close your browser you also lose the information.
4. Cookie store limited amount of data i.e. 4Kb(4096 bytes)	4. It stores unlimited amount of data
5. Cookies can only store string	5. Store user's object in session
6. Less secure	6. More secure
7. setcookie() function is used to create cookie	7. session_start() functions is used to create session